

RACING DESTRUCTION SET:
SCALEXTRIC EN PANTALLA

SEMANAL

CHISPAS.
LOS MEJORES
TRUCOS PARA
COMMODORE

ENSAMBLADOR Y CODIGO MAQUINA: TAL PARA CUAL

PROGRAMAS TOTALMENTE COMPATIBLES



Director:
ANTONIO M. FERRER ABELLO

Redactor-jefe:
FERNANDO LOPEZ MARTINEZ

Redacción:
ANTONIO CARVAJAL
JUAN M. LOPEZ MARTINEZ
PABLO GARCIA MOLINA
JOSE LUIS DE DIEGO
IGNACIO BARCO LUENGO
ALFREDO SINDIN VALERO

Colaboradores
JOSE LUIS VAZQUEZ DE PRAGA
ANTONIO MANZANERA

Secretaría de Redacción:
PILAR MANZANERA AMARO

Diseño y Maquetación:
CARLOS GONZALEZ AMEZUA
MAITE CONDE

Ilustraciones:
ANTONIO PERERA
RAMON POLO

Fotografía.
Equipo Gálata

Directora Publicidad
CARMINA FERRER
Tel. 457 69 23

Publicidad Barcelona:
ISIDRO IGLESIAS
Avda. Corts Catalanes, 1010
Tel (93) 307 11 13

Director de Producción:
VICENTE ROBLES

Directora de Administración:
MARIA ANTONIA BUITRAGO

Suscripciones:
MARIA GONZALEZ AMEZUA

Redacción, administración, publicidad y suscripciones
Plza. República Ecuador, 2
28016 MADRID. Tel. 250 58 20
Télex 49371 ELOC E

Dirección para correspondencia:
Aptdo. de Correos 61.294
28080 MADRID

Fotocomposición:
VIERNA, S. A. MADRID

Fotomecánica:
RODACOLOR, S. A. MADRID

Imprenta:
GRAFICAS REUNIDAS, S. A. MADRID

Distribución:
COEDIS. Valencia, 245
BARCELONA

Precios para España. Ejemplar: (160 ptas. (IVA incluido). 155 ptas. (Canarias, Ceuta y Melilla). (La suscripción anual incluye 52 números.)
Distribución Cono Sur:
CADE, S.R.L.
Pasaje Sud América, 1532
Tel. 21 24 64.
Buenos Aires 1.290. Argentina.

Depósito Legal. M. 40920-1985.
Impreso en España.

SUMARIO

Como ya anunciamos anteriormente, es nuestra intención que en TU MICRO COMMODORE tengan cabida todos los usuarios de los microordenadores Commodore, y no sólo los afortunados poseedores de un C-64. Buena muestra de ello es el número que esta semana ha llegado a vuestras manos, repleto de «compatibilización», desde la sección de TECLA A TECLA hasta el SOFTWARE. Sin duda la idea os atrae; pues bien, en esta línea continuaremos, aunque en ocasiones no será posible, como bien sabéis, por mucho que lo intentemos, compatibilizar completamente todos los programas, pues ello nos obligaría a no sacar todo el partido posible de alguno de los aparatos. Seguros de vuestra comprensión, nos despedimos hasta la próxima semana.





NOTICIAS

EDITORIAL NORAY CON COMMODORE 64. CINCO LIBROS ESPECIALIZADOS EN NUESTROS ORDENADORES

Dentro de su colección de libros de informática, la editorial Noray ha tenido la buena idea de incluir cinco títulos destinados única y exclusivamente a los ordenadores Commodore.

Dentro de los mismos, podremos encontrar aplicaciones muy diversas, desde la enseñanza de los más pequeños, hasta la gestión para la casa y el pequeño negocio.

Sus títulos son los siguientes:

— Comodore 64. Qué es, para qué sirve y cómo se usa.
— Commodore 64. Aplicaciones prácticas para la casa y los pequeños negocios.
— Prontuario del Commodore 64.
— 18 juegos dinámicos para tu Comodore 64.
— El Commodore 64 y los niños

LIBROS DE INFORMATICA

EDITORIAL NORAY

COMPLEMENTO A NUESTRO ARTICULO SOBRE JOYSTICKS APARECIDO EN EL NUMERO 7

La firma Serma, que tanto hace por nuestra diversión con sus magníficos programas, completa ahora el ciclo de todo juego distribuyendo también el periférico de juego por excelencia: el joystick.

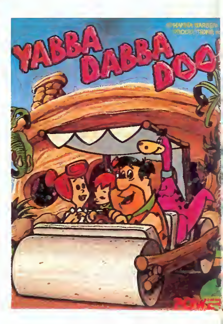
Se trata del Joycard de Hudson Soft, un original joystick plano con dos botones de disparo, compatible con Commodore 64 y VIC-20.

La nueva tecnología al alcance de todos los «juegomanos» gracias a Serma.



POWER Y LOS PICAPIEDRAS

La casa de software barcelonesa POWER ha lanzado al mercado su última novedad «YABBA DABBA DOOO!, el característico grito de euforia del famoso personaje de dibujos animados Pedro Picapiedra. El juego se desarrolla en el fantástico mundo de los hombres del Neardental, con los troncomóviles, los dinosaurios domésticos como el simpático Dino y toda suerte de animales y artilugios que siempre acompañan a Pedro y a su familia, todo amenizado con una graciosa musiquilla, y todo esto sólo por... 1.800 pesetas.



chispas

CHISPA

Si deseas emplear números aleatorios en un programa, has de saber que tales números pueden no ser tan aleatorios. Algunos equipos incorporan la función RANDOMIZE que desordena el valor inicial de la semilla a utilizar internamente en su sistema de obtención. Para simular este comando, podemos usar: **X=RND(TI/13).** El 13 es un número primo que asegura un desorden mayor en la semilla.

CHISPA

Si quieres utilizar en tus programas las teclas: **CTRL, COMMODORE** y **SHIFT,** las puedes encontrar en la posición 653; para que pruebes tu mismo el valor de cada una, ejecuta la siguiente línea de programa:

```
10 PRINTPEEK (653):GO
TO10
```

Verás aparecer sus valores al pulsar la tecla correspondiente.

CHISPA

Para cargar un programa desde la unidad de disco, normalmente listamos el directorio y, delante del programa deseado, tecleamos **LOAD**; a continuación saltamos con el cursor hasta la última comilla del nombre, y tecleamos **,8:**, o bien, **,8,1:**.

Pensemos en lo útil que sería tener estos caracteres incluidos en el directorio; pues bien, este efecto se puede conseguir de la siguiente manera:

Supongamos que nuestro programa se llama «programa», veamos los dos casos:

```
,8:  SAVE"PROGRAMA[S
HIFT ESPACIO][COMMO
DORE D][8][SHIFT[2]",8
,8,1 SAVE"PROGRAMA[S
HIFT ESPACIO][COMMO
DORE D][8][COMMODO
RE D][1]",8
```

Cada término entre corchetes ocupa un caracter, así pues, la longitud del nombre se limitará consecuentemente.

CHISPA

Si queremos cargar el directorio de un disco, pero sólo quieres ver un determinado tipo de ficheros, utiliza: **LOAD "$*=[letra]",8** donde [letra] se puede sustituir por: P; para obtener todos los programas. S; para obtener ficheros secuenciales. U; ficheros «user» y R; ficheros relativos.

También se puede utilizar con los demás «wildcards». Ej.: **LOAD "Sc*=p",8.** Cargará todos los programas que comiencen por la letra **C.**

Si quieres saber el nombre de un disco, su identificación o el número de bloques libres; pero no te interesa cargar todo el directorio, puedes emplear el nombre de un programa ficticio, por ejemplo: **LOAD"$:aSdoxxz", 8** y **LIST.**

CHISPA

En chispas anteriores, hemos visto instrucciones para detener el programa hasta que se pulse una tecla: **POKE198,0:WAIT198,1.** Pero para el que esté interesado en detener el programa hasta que se deje de presionar una tecla (nunca se sabe) aquí tiene cómo hacerlo: **WAIT203,64.**

CHISPA

Hay programas que pueden funcionar perfectamente en un C64 o en un VIC-20, el único problema que podemos encontrar, será el del tamaño de la pantalla. Antes de cada programa de este tipo, podemos ejecutar **SYS65517,** lo cual sitúa el número de columnas en el registro X, y el número de filas en el registro Y; estos registros los encontraremos en las posiciones 781 y 782 respectivamente.

CHISPA

Para aumentar la velocidad de los programas en BASIC, no se puede hacer mucho, pero si vas a utilizar un valor muy a menudo, asígnalo a una variable. Así por ejemplo, **POKE53280,1** es más lento que **POKEA,1;** habiendo efectuado anteriormente **A=53280.** Esto es debido a que le es más fácil para el ordenador, buscar en la zona de variables la A, que traducir el entero 53280 y trabajar con su valor.

CHISPA

La zona de memoria del BASIC tiene un par de punteros que señalan su inicio (43,44) y su final (55,56). Hay veces en que es necesario delimitar esto nuevamente a una dirección diferente (2049-40960). Para hacerlo, veamos un ejemplo.

Supongamos que la nueva dirección es la variable **N** (tanto para el inicio como para el final). Primero debemos extraer el byte alto (BA) y el byte bajo (BB) de esta dirección:

```
BA=INT (N/256)
BB=N-BA*256
```

Una vez obtenidos estos, ajustaremos el inicio:

```
POKE43,BB:POKE44,BA:
POKEN-1,0:NEW
```

O el final...

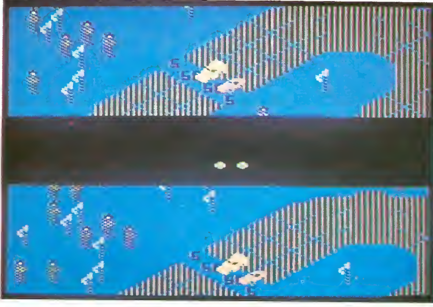
```
POKE55,BB:POKE56,
BA:CLR
```

Tengamos en cuenta que deberemos sustituir **N** por nuestro nuevo valor, y que el inicio deberá ser menor que el final.

RACING DEST

Dos pilotos calientan los motores de sus máquinas en la línea de salida, mientras el semáforo pasa del rojo al amarillo y por fin... ¡al verde! RACING DESTRUCTION SET, el legendario juego del Scalextric llevado a la pantalla de un ordenador Commodore.





En 1447 se exhibió por las calles de Meiningen (Alemania) un coche sin caballos accionado mediante una combinación de palancas y engranajes. Desde la creación de este antepasado del automóvil hasta perfeccionar la técnica de la Fórmula 1, el campo de la automoción ha dado pasos gigantescos.



RUCTION SET

Paralelamente al desarrollo de la velocidad de los primeros bólidos, se iban gestando las grandes competiciones de velocidad mundiales. Gracias a la labor de Rick Koening, Connie Goldman y Dave Warhol (Artistas Electrónicos), podemos tomar parte activa en este deporte.

RACING DESTRUCTION SET es un duelo de velocidad donde los jugadores (dos humanos, o un humano y el ordenador) pueden construir su propia pista y elegir un vehículo idóneo, de entre los diez existentes, para correr en la misma. Al final, el objetivo a cumplir por el usuario es el mismo sean cuáles sean las condiciones escogidas: atravesar primero la línea de meta.

Ahora bien, según la reglamentación que rija la carrera, un corredor variará su misión añadiendo cometidos suplementarios a esta. Por ejemplo, en un Rallye de destrucción para el nivel Severo, aniquilar el coche del oponente, o en un circuito de velocidad, ir superando el record de tiempo.

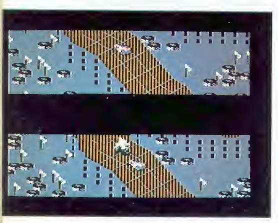


SELECCIONANDO LAS OPCIONES DE LA CARRERA PARA DESTRUIR COCHES A PLACER

El juego ocupa más de 300 Kbytes de memoria y dada la capacidad de 64 Kbytes del Commodore 64, requiere ir extrayendo sucesivamente las partes del programa utilizadas en cada instante. Para acceder a todas las posibilidades que se nos ofrecen, el ordenador nos irá indicando mediante mensajes mostrados en la pantalla cuál es la cinta de casete a introducir en el datassette.

Aparece aquí un problema debido a la dificultad para distinguir las cuatro caras de las cintas de juego. La cinta de color negro corresponde a las caras 1 y 2, distinguiendo la cara número 1 en fun-

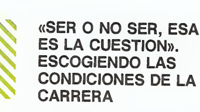
ción de la cabecera de la misma, cuyo lema es CAR RACING DESTRUCTION SET. Respecto a las caras números 3 y 4, de color amarillo, sólo es posible la distinción realizando una prueba una vez cargado el programa. Después, la mejor manera de recordar dónde está situada cada cara consiste en hacer una pequeña marca sobre las caras 1 y 3, a título orientativo.



Al comenzar la carga de la cara número 1, una pantalla publicitaria va mostrando los éxitos de DROSOFT en el campo del software para Commodore al compás de la música de Aviador Dro; a continuación aparece la presentación del programa, para acabar cargando los menús de opciones.

Existen cinco menús de este tipo, preparados para satisfacer las exigencias de los corredores (excluyendo el apartado económico, lo cual resta emoción a la carera): Menú Principal, Menú de Carrera, Menú Nivel y Opciones, Menú Montaje de Pistas y Menú Montaje de Vehículos.

Si nuestro pulso se acelera y ya hemos puesto cara de velocidad, lo mejor es empezar a correr inmediatamente en la pista de demostración, donde dos CAN-AM se disputarán la victoria a dos vueltas sin demasiadas dificultades. En caso contrario, la lectura de los siguientes párrafos subsanará cualquier dificultad relativa a la construcción de pistas o a la selección de vehículos.



«SER O NO SER, ESA ES LA CUESTION». ESCOGIENDO LAS CONDICIONES DE LA CARRERA

El Menú Principal o Gran Menú es el primer obstáculo antepuesto entre nosotros y la pista de carreras. Aquí, como en las restantes pantallas de selección, las cuatro direcciones principales del joystick situarán el cursor sobre la opción escogida y el disparo hará definitiva la elección. El Gran Menú sirve para indicarle al Ordenador el número de corredores humanos participantes (1 ó 2); una vez hecho esto, conviene pasar al Menú de Carrera mediante Opciones y Carrera.

En este Menú podemos modificar nuestro vehículo o la pista y establecer el grado de dificultad idóneo con el Menú de Nivel y Opciones, o bien regresar al Menú Principal para acondicionar de nuevo el número de jugadores.

En el Menú de Nivel y Opciones escogemos tanto el Nivel como las condiciones de la carrera. El nivel admite tres variantes: Normal, Avanzado y Severo. El primero prolonga la vida de los bólidos reduciendo notablemente los daños producidos por las colisiones; en el segundo, los choques mermarán la velocidad de los vehículos, llegando a detenerlos totalmente; y bajo el Nivel Severo, recibir un impacto reduce mucho más la potencia de los bólidos, y estos serán destruidos si salen (¡o son sacados!) del circuito. Para los novatos, lo indicado es el Nivel Normal, pero lo verdaderamente divertido se halla en el Nivel Severo.

Las opciones abarcan aspectos muy diversos tales como la velocidad máxima del coche amarillo del ordenador en las rectas (20 millas por hora menos para un Novato, 10 menos para un Exper-



to y máxima para un Profesional). Realmente, en vista de la falta de categoría del Ordenador para la conducción, carece de aliciente (y profesionalidad) no declararse Profesional.

También es posible remarcar si se trata de una Carrera o de un rallye de Destrucción. En este último caso, podremos utilizar las armas incorporadas a los bólidos (minas, litros de aceite y trituradores); en una prueba de velocidad, el armamento incorporado supone un estorbo.

La gravedad elegida introduce mayor variedad a la hora de competir. Catorce planetas corresponden a catorce valores distintos de gravedad, desde la de Júpiter (máxima) hasta la de la Luna (mínima y lunática). A menor gravedad, los saltos serán más espectaculares y los coches colisionarán contra el suelo con menor frecuencia. Las condiciones ideales son las de la Luna, Mercurio o Io, dependiendo de las características del Circuito.

Un corredor se alza con el triunfo si atraviesa la línea de meta entre 1 ó 9 veces, según lo escogido. Por otra parte, si un vehículo es incapaz de seguir corriendo, el otro es declarado vencedor en el instante mismo en el que cruce la línea de meta.

Al margen de la carrera en sí y en un plano decorativo, cuatro paisajes aparecen al lado de la carretera, lo cual produce un efecto óptico distinto en cada caso: Track no permite vislumbrar la pista con claridad, Motox está especialmente indicado para monitores en color, Abstract y Lunar tienen su función en monitores dicromáticos.

CEMENTO, LADRILLOS Y MUCHA IMAGINACION... ¡CONSTRUYE TU PROPIA PISTA!

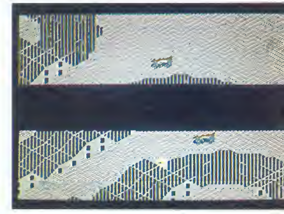
La tercera línea del Gran Menú y la cuarta del Menú de Carrera conducen al submenú de Montaje de Pistas, en el cual vislumbramos siete funciones distintas, cada una de ellas destinada a una tarea particular.

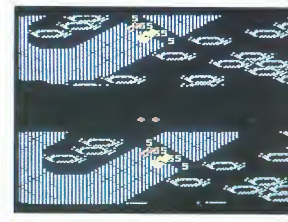
Cargar una pista equivale a extraer de una de las casetes del juego o de una propia, un circuito previamente grabado. El programa consta de un total de 51 recorridos distintos que, sumados a todos los que tu puedas crear, dan un total de infinitos recorridos. Algunos de ellos son muy difíciles (Tiger, Variety, Killer o Spiral), otros demasiado fáciles (Destruction o Headon), curiosos (Jumps o Whichway), y los más son reproducciones de circuitos de competición del mundo entero (Jarama, Zolder, Silverstone, Mónaco, etc.).

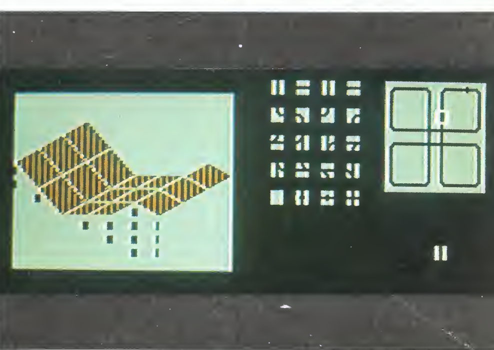
Un circuito creado puede ser almacenado para sucesivas carreras en una cinta mediante la opción de Grabar, si antes se ha Formateado la Cinta. Utilizando una casete de 10 minutos de duración para grabar las pistas en la cara número 1 y los vehículos en la número 2, ahorraremos mucho tiempo.

Uno de los grandes atractivos de RACING DESTRUCTION SET es su editor de pistas (Montar/Editar pista), el cual, contrariamente a lo expuesto en las instrucciones, tiene un manejo complicado.

Disponemos de 20 piezas para







diseñar un circuito, distribuidas en cinco grupos de cuatro de arriba a abajo. Todas ellas permiten variar el terreno (asfalto, tierra o hielo), la altura y el ancho de ese tramo de pista (1 a 3 cuadros, exceptuando la línea de salida/meta).

La primera fila muestra a su izquierda dos piezas denominadas saltos en virtud de la posibilidad de ir alterando la altura de cada cuadro, creando así elevaciones y depresiones dentro del mismo tramo. A la derecha, los dos chicanes admiten variaciones sobre el ancho de la pista también en cada cuadro.

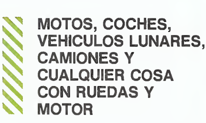
La segunda contiene cuatro giros; la tercera y cuarta muestran los desvíos, y en la última están dispuestas cuatro piezas especiales: 1 intersección, 2 salida/meta y una pieza en blanco, a modo de goma de borrar. La línea de salida/meta admite cuatro variantes: una sin coches, una con ambos vehículos en el mismo sentido, y otras dos con los dos bólidos en sentido contrario.

En cuanto al procedimiento de construcción, el primer paso es la extracción de una pista adecuada para realizar modificaciones sobre ella, situando el cursor sobre la pieza escogida y pulsando el botón de disparo. Los mecanismos de ensamblaje de piezas son de dos tipos principalmente: sobre una pieza del cuadro o del circuito.

Para alterar una pieza del cuadro basta seleccionar esta y realizar las modificaciones deseadas, colocando después el resultado en el cuadro del circuito. Cuando la «memoria» (parte derecha del editor) está ocupada por un tramo de recorrido, el editor colocará sobre el cuadro de pista esta pieza, a menos que sea borrada al situar el cursor sobre un cuadro negro.

Para los tramos contenidos en el cuadro del circuito, lo preceptivo es situar el cursor en un cuadro negro despejando así la «memoria», y colocar el cursor sobre el tramo de pista que deseemos variar, lo cual nos introduce en los menús de alteraciones. Cuando hayamos terminado depositaremos el tramo donde nos interese.

Además, existe la posibilidad de borrar todo el recorrido y dejar el cuadro en blanco (muy desaconsejable), para comenzar de nuevo la tarea. Cuando esta finaliza, la opción Exit chequeará la pista y mostrará las piezas que no estan bien situadas (los errores más comunes son los relativos a la anchura y altura de los diversos tramos), o seguirá el curso normal del programa si el circuito ha sido construido correctamente.



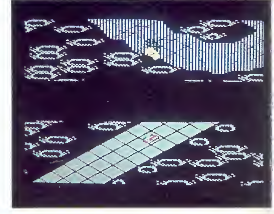
MOTOS, COCHES, VEHICULOS LUNARES, CAMIONES Y CUALQUIER COSA CON RUEDAS Y MOTOR

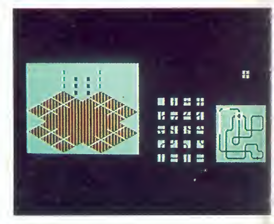
Ya sentimos el alivio de haber dejado atrás las complicaciones de la construcción de pistas y nos enfrentamos con una nueva etapa, sencilla pero decisiva, la elección de un vehículo apropiado para correr en el circuito bajo las condiciones marcadas con anterioridad.

El catálogo de transportes es bastante completo e incluye todo tipo de máquinas: Can-Am, Jeep, Rover Lunar, Moto de Cross, VW Cucaracha, Trailer, Sting Ray, Coche usado, Moto de Ciudad e Indy/Gran Prix; aunque realmente sólo aquellas con velocidad elevada son dignos de mención (Cam-Am de 5000 CC, Sting Ray de 8200 CC, Indy/Gran Prix de 3000 CC, Coche Usado de 6600 y Moto de Ciudad de 1000 CC).

Cada vehículo admite una serie de variaciones que comprenden desde el tipo de neumáticos utilizados: lisos (idóneos para asfalto), de calle (asfalto y tierra), de clavos (hielo) o de tacos (tierra); hasta la cilindrada (CC). Otras posibilidades son las minas, el aceite, los blindajes y trituradores. Cualquier alteración sobre la disposi-





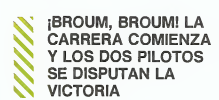






ción inicial repercute en su peso (decisivo en los saltos), velocidad máxima alcanzable, aceleración (tiempo empleado en llegar a la velocidad máxima) y resistencia a los golpes.

Un gran coche preparado para los Rallyes de destrucción es un Can-Am de 55000 CC con neumáticos lisos y 9 litros de aceite. Otra opción muy destacada es la de la Moto de Ciudad, la cual destaca por su gran estabilidad, aunque también ofrece un pequeño inconveniente: si se queda atascada en una subida, perderá la carrera irremisiblemente.



¡BROUM, BROUM! LA CARRERA COMIENZA Y LOS DOS PILOTOS SE DISPUTAN LA VICTORIA

Hemos elegido/modificado un circuito hasta que se ha ajustado a nuestras exigencias; de igual modo, hemos manipulado a placer las condiciones de la carrera y escogido un vehículo idóneo, ahora sólo pensaremos en ganar.

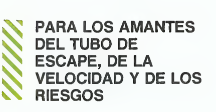
El semáforo pasa del rojo al amarillo y después al verde, momento en el cual los coches aceleran con el joystick hacia delante y lanzan minas y aceite con el botón de disparo. Cada corredor observa su pantalla y la del contrario esperando conseguir una ventaja decisiva para alzarse con el triunfo. En la primera curva, un coche ha girado utilizando las direcciones laterales del joystick y pierde terreno porque el otro ha girado con las diagonales superiores del mismo. Además, el primero confundió la dirección de la curva y giró a la derecha del piloto, en lugar de a la izquierda. Mientras este recupera el terreno cedido, el anterior se acerca al primer salto y reduce su velocidad con el joystick hacia atrás para no chocar. El resultado será imprevisible y seguramente sorprendente.

No es todo tan simple como pueda parecer en un principio, ya que cada corredor tiene un estilo propio de vencer (o de perder, si este estilo es pésimo), conforme a una serie de estrategias ideadas previamente.

En una curva recuerda que basta un giro para que el coche termine con éxito esta maniobra; los demás movimientos provocarán sólo pérdidas de tiempo, reducibles al máximo cuanto más nos acerquemos a la cara interna de la curva. Estos tramos están especialmente indicados para ponerle las cosas difíciles a tu oponente echando aceite sobre el terreno.

Los saltos deben ser recorridos con una velocidad adecuada, ni muy rápido ni muy despacio, sino todo lo contrario. Las primeras vueltas te servirán de tiempo de reconocimiento, para saber la forma que tiene la pista y dónde se encuentran los obstáculos más peligrosos; al igual que el ordenador comprueba sus frenos, cuando en realidad está comprobando los saltos existentes en el terreno, antes de empezar a correr.





PARA LOS AMANTES DEL TUBO DE ESCAPE, DE LA VELOCIDAD Y DE LOS RIESGOS

POLE POSITION y PIT STOP I Y II son juegos de competición entre Fórmula 1 a lo largo de algunos cricuitos. RACING DESTRUCTION SET, supera totalmente esta concepción de los Rallyes por ordenador, pues se trata de una emulación de juguetes tan conocidos como el Scalextric, TCR y otros.

No sólo desarrolla la competitividad entre jugadores sino que además nos brinda mucha diversión bajo unos gráficos no muy lúcidos, pero con algunos aspectos destacables (saltos y vueltas de campana).

El principal inconveniente adherido al juego es el tiempo de carga de todas las opciones y subopciones. De forma general, son necesarios 30 minutos para jugar con una pista autoconstruida y unos vehículos seleccionados por nosotros mismos.

Cabe felicitar a la Asociación de Artistas Electrónicos por la creación de este juego (muy, muy original en su género) y a Dro Soft, que con el buen hacer que le caracteriza, ha traducido las instrucciones y los mensajes de la pantalla.

LOS 7 MAGNIFIC

Colaboremos todos en la confección de estos 7 MAGNIFICOS de TU MICRO COMMODORE. Envíanos el nombre de tu programa favorito, dejando muy claro tu nombre y dirección. Todos los meses sortearemos una suscripción por un año a nuestra revista entre las respuestas recibidas. Anímate y escríbenos a: TU MICRO COMMODORE. (7 MAGNIFICOS). APARTADO DE CORREOS 61.294. 28080 MADRID.

2. COMMANDO



3. ONE ON ONE

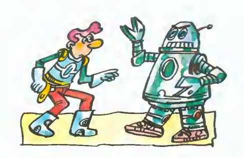




4. WIZARDRY



5. IMPOSSIBLE MISSION



N.º 1 ESTA SEMANA

RAMBO

Armado de una cámara fotográfica, su espectacular preparación física y poco más, Rambo, el conocido protagonista del film Acorralado, regresa a la selva vietnamita, con la única misión de encontrar pruebas que disipen las dudas del Congreso de los Estados Unidos, sobre la existencia de prisioneros americanos en manos del Ejército Rojo aún en nuestros días. La misión en principio sencilla para un hombre como Rambo, se complica cuando descubre la triste realidad: todavía existen camaradas suyos sufriendo cautiverio en un campo de concentración norvietnamita. Ya no basta con obtener una prueba: es necesario rescatarlos.



6. SKYFOX

7. WINTER GAMES



ORDENACION RAPIDA DE MATRICES

El tema que vamos a tratar hoy, en nuestro apartado de software, puede considerarse un clásico de la informática de siempre; lo cual indica que la resolución de este problema es bastante común en aplicaciones «serias».

Para abrir boca, vamos a plantear el problema. Como todos sabemos, una matriz es un conjunto de valores reagrupados bajo un mismo nombre, donde el acceso a cada uno de ellos, se realiza a través de un índice representativo de su posición dentro de la matriz. Una matriz se define normalmente por medio de una sentencia **DIM,** en la cual se especificará el nombre de la matriz y su tamaño. Existen muchos tipos de matrices atendiendo a su dimensión, y según esta, pueden clasificarse en unidimensionales (vectores), bidimensionales (tablas), etc... Para comprender esto imaginemos las dimensiones de nuestro espacio, línea, plano, triedro... etc.

Por lo general, las más usadas por su relativa facilidad de uso, son las matrices unidimensionales; y lo que hoy vamos a ver, es la clasificación de los elementos de una de estas matrices en un orden determinado (ascendente o descendente).

El campo de aplicación de esta técnica es de lo más variado, siendo las necesidades de cada uno las que determinen su uso, pero por citar algunos: ordenación de puntuaciones de un juego, clasificación de elementos de un fichero, ordenación de equipos de fútbol, las notas de una clase, etc.

Es muy común encontrar diversos métodos de ordenación comparados entre sí, pero siempre hay alguno que destaca sobre los demás, así pues, atendiendo a una necesidad práctica, esta vez iremos al grano y describiremos uno de los métodos más rápidos: el SHELL-METZNER. La base matemática de este método es relativamente compleja, así que nos limitaremos a su empleo sin entrar en explicaciones matemáticas que escapan del propósito general de esta publicación.

Como base comparativa, hay que decir que este método es del orden de 10 veces más rápido que sus homólogos «facilillos», entre los que se encuentra el conocidísimo BUBBLE SORT.

El listado del programa que adjuntamos a título ilustrativo consta de varias partes: creación de la matriz (de forma aleatoria), impresión de sus elementos desordenados, ORDENACION e impresión de sus elementos ordenados. Este listado trabajará perfectamente en cualquier ordenador Commodore, para los usuarios del C64 y C128 en modo 64 sirve tal como está, mientras que los poseedores del C16 y PLUS 4 tendrán que cambiar la línea 100, para que quede como sigue:

```
100 N=80:O$="00":COLOR4,1:
COLOR0,1:COLOR1,6:
DIMX(N):GOSUB180
```

La parte principal del programa es la subrutina de ordenación que se encuentra a partir de la línea 190 (190-220).

Es fácilmente utilizable como subrutina de nuestros propios programas. Habrá que tener cuidado con las variables que se emplean en ella, para que no estén duplicadas en la parte principal del programa.

Por último, diremos que la ordenación se puede realizar en cualquier sentido, es decir, ascendente o descendente; en el ejemplo, se realiza en orden ascendente; para efectuar la ordenación en sentido descendente, habrá que cambiar el signo mayor que (>) de la línea 200 por un menor que (<).

Que usted lo ordene bien.

```
100 N=80:O$="00":POKE53280,0:POKE53281,0:POKE646,5:DI
MX(N):GOSUB180                                     -137-
110 PRINT"{CLR}ESTA ES LA MATRIZ DESORDENADA":PRINT"<
29 T>"                                             -091-
120 FORI=1TON:X$=STR$(X(I)):X$=MID$(X$,2):X$=LEFT$(O$
,3-LEN(X$))+X$                                     -150-
130 PRINTX$;" ";:NEXT:TI$="[6 0]"                  -172-
140 GOSUB190:PRINT:PRINT"MATRIZ ORDENADA EN";INT(TI/6
0+.5);"SEGUNDOS!!"                                 -233-
150 PRINT"<32 T>"                                  -231-
160 FORI=1TON:X$=STR$(X(I)):X$=MID$(X$,2):X$=LEFT$(O$
,3-LEN(X$))+X$                                     -154-
170 PRINTX$;" ";:NEXT:END                          -037-
180 FORI=1TON:X(I)=INT(RND(1)*N*5):NEXT:RETURN
                                                   -189-
190 W=INT(LOG(N)/LOG(2)):FORQ=WTO1STEP-1:H=2^(Q-1):FO
RJ=H+1TON:I=J-H:T=X(J)                             -220-
200 IFT>=X(I)THEN220                               -197-
210 X(I+H)=X(I):I=I-H:IFI>=1THEN200                -068-
220 X(I+H)=T:NEXT:NEXT:RETURN                      -182-
```

CURSO DE BASIC

Hemos visto hasta ahora dos claros ejemplos de la forma en que podemos asignar valores a las variables. El primero de ellos, consiste en hacerlo por medio del símbolo de igualdad (=), dentro del propio programa; el segundo, desde fuera de este, por medio de la sentencia **INPUT.**

Además de estos dos métodos, existe un tercero que permite incluir determinada cantidad de datos dentro del propio programa, para su posterior tratamiento. Este sistema especial de almacenamiento recibe el nombre de **DATA** (datos).

Para comprender bien el significado de esta sentencia es necesario saber cómo trabaja el intérprete del BASIC con las líneas de programa. De momento, sabemos que van siendo interpretadas por este desde el número de secuencia más bajo en adelante y siguiendo, en todo caso, las indicaciones condicionales o imperativas desencadenadas por las posibles bles sentencias **GOTO** incluidas en el texto BASIC.

Sin embargo, podemos determinar «zonas», dentro de la vasta extensión del texto BASIC, destinadas a albergar datos que puedan ser recuperados, dentro de programa, por medio de una sentencia especial de lectura **READ** (leer), de forma similar a como la sentencia **INPUT** es capaz de admitir datos desde el exterior.

Las sentencias **DATA** califican una o un grupo de líneas como zona de datos, de forma que son consideradas como tales por el intérprete del BASIC. Por ello, no son tratadas en la forma que las demás líneas de texto sino que comportan una especie de «almacén», disponible para ser leído de forma secuencial, es decir, una por una; desde cualquier punto de programa en el que se encuentre una sentencia **READ,** no suponiendo, sin embargo la ejecución de ningún otro tipo de acción.

LA SENTENCIA DATA

La palabra BASIC **DATA** debe encabezar cada línea destinada a contener un dato, o una lista de ellos si los separamos por comas (,), de forma similar a lo que sería el **INPUT** de una serie de variables. Por ejemplo, podemos incluir en **DATA** las iniciales de los meses de la forma:

10 DATA ENE,FEB,MAR,ABR, MAY,JUN,JUL,AGO,SEP,OCT, NOV,DIC

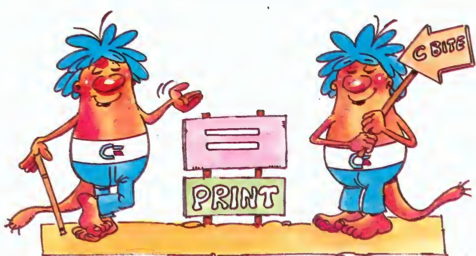
Aunque podríamos optar también por incluir un solo elemento por instrucción, empleando la forma:

10 DATA ENE

....

130 DATA DIC

En nuestro ejemplo, hemos confeccionado una lista de datos alfanuméricos sin necesidad de emplear las comillas (OFF)**).** Esto se



Para asignar valores a las variables, podemos hacerlo bien por el signo de igualdad o por medio de la sentencia PRINT.



Existe un tercer método para incluir una determinada cantidad de datos dentro del propio programa, a través del mantenimiento de DATA.

debe a que el tipo de literales que empleamos no incluye caracteres en minúsculas, símbolos gráficos ni carácteres especiales como la propia coma (,) o los dos puntos (:).

Si deseamos incluir caracteres de este tipo dentro de las sentencias **DATA,** estamos obligados a acotar los elementos entre comillas, de forma que no pueda producirse confusión al efectuar su lectura desde el programa BASIC, por medio de la sentencia **READ.** Esta es la forma correcta de escribir la lista de meses si deseamos emplear mayúsculas y minúsculas:

10 DATA "Ene","Feb","Mar"," May","Jun","Jul","Ago","Sep", Oct","Nov","Dic"

De igual manera, este sistema nos permite la inclusión de signos de puntuación y caracteres gráficos, como el de subrayado obtenido por la pulsación de la tecla COMMODORE y la de la T:

10 DAT "RODRIGUEZ PEREZ, Emilio", "TTTTTTTTTT TTTTTTTTTTTTT"

Las sentencias **DATA** pueden incluirse en cualquier parte del programa, aunque es norma casi generalizada el incluirlas inmediatamente antes o después de la sentencia BASIC que ordena su lectura, o bien formando un bloque compacto al final del programa.

Por otro lado, una **DATA** permite almacenar un valor numérico válido para asignar a una variable de este tipo, es decir, cualquier entero o real, expresado tanto en el formato convencional como en notación científica. Son valores válidos, por ejemplo, los siguientes:

Las sentencias DATA pueden incluirse en cualquier parte del programa, aunque normalmente se hace antes o después de la sentencia BASIC que ordena su lectura.



10 DATA 12345,7546.34,0,12E−23

No pueden emplearse, sin embargo, nombres de variables ni expresiones matemáticas de ningún tipo que vinculen a valores variables ni literales fijos, como por ejemplo:

10 DATA C,B*4,5/50

Realmente, se trata de las mismas limitaciones en la forma de introducir los datos que si de la sentencia **INPUT** se tratara y, la razón de todo ello, es que en las



Las sentencias DATA califican una o un grupo de líneas como zona de datos.

sentencias de entrada de datos, el BASIC se limita a admitir directamente el valor que se suministra a la variable de forma directa, sin entrar en la interpretación que realizaría, por ejemplo, si se tratara de una asignación por medio del símbolo de igualdad.

Por ello, no es posible suministrar como **DATA** una expresión que forzosamente deba ser evaluada, como por ejemplo una ecuación...

19 DATA 2X ↑ 2–6X+5



... y tratar de ejecutar una sentencia **READ Y** en el texto BASIC. Si deseamos obtener el valor de la variable **Y** en función del de **X,** estamos obligados a incluir la expresión en la forma conocida que emplea el símbolo de igualdad, es decir:

10 Y=2X ↑ 2–6X+5

LA SENTENCIA READ

Del mismo modo que la colocación de las sentencias **DATA** es arbitraria, las correspondientes **READ** pueden ubicarse en el lugar del programa donde sea necesario efectuar la lectura del siguiente dato a procesar. A cada

La palabra BASIC DATA debe encabezar cada línea destinada a contener un dato, o una lista de ellos si los separamos por comas.

sentencia **READ** debe acompañar, necesariamente, el nombre o nombres de las variables a las que se desea aignar valores, separadas por comas (**,**), si son varias, de forma similar a lo comentado para **INPUT.** El formato general de la sentencia **READ** es, por lo tanto:

READ variable [,...,variable]

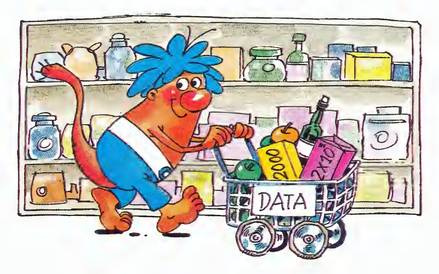
Donde, de existir más de una variable, debe ponerse cuidado en que se encuentren colocadas dentro de las **DATA** en la forma adecuada, en evitación del error de lectura **TYPE MISMATCH ERROR** (error en la clase de datos) que se producirá en caso contrario.

El siguiente programa de ejemplo va leyendo las **DATA** antes propuestas y asignando sus valores a la variable **X$,** imprimiéndose sus resultados en pantalla:

```
10 FOR I=1 TO 12
20 READ X$
30 PRINT X$,,
40 NEXT
50 DATA ENE,FEB, MAR, ABR,
MAY,JUN,JUL,AGO,SEP,OCT,
NOV,DIC
```

En este caso, hemos leído la data por medio de un bucle **FOR NEXT** por darse la circunstancia de que conocemos, de antemano, el número de elementos que la **DATA** contiene.

Puede ocurrir que deseemos



Una DATA permite almacenar un valor numérico, expresado tanto en formato convencional como en notación científica.



A cada sentencia READ debe acompañar el nombre de las variables a las que se desea aignar valores, separados por comas.

que el número de datos sea variable, por ejemplo, para calcular la media aritmética de determinada serie de valores. En este caso, nos vemos obligados a marcar el final de los datos por medio de algún valor especial, un número negativo, por ejemplo, que nos permita detectar el final de estos.

Esto es debido a que cuando la sentencia **READ** trata de efectuar una lectura sin quedar ya datos que capturar, se produce el error **OUT OF DATA ERROR** (error por final de los datos). El siguiente programa de ejemplo calcula la media de los números que introduzcamos en las líneas **DATA** del final:

```
10 READ X
20 IF X<0 THEN 50
30 E=E+1:T=T+X
40 GOTO 10
50 M=T/E
60 PRINT "TOTAL ELEMEN TOS:"T
70 PRINT
80 PRINT "VALOR MEDIO...: "M
90 DATA 24,92,4,0,33,12,9,65,38,40,76,6,−1
```

La línea 20 tiene como única misión detectar el final de los datos,- calculando la media e imprimiendo los resultados obtenidos. Mientras que esto no suceda, cada elemento es sumado al contador **E,** y cada nuevo valor acumulado en la variable **T,** lo cual permite calcular la media aritmética empleando la fórmula de la línea 50.

LA SENTENCIA RESTORE

Como ya hemos dicho, la lectura de las **DATA** se produce elemento a elemento hasta llegar a su final. Para ello, el BASIC se sirve de un puntero o indicador, que recuerda en todo momento el elemento siguiente a leer y en qué número de instrucción se encuentra.

La sentencia **RESTORE** permite posicionar este puntero al comienzo de la primera **DATA,** para que las sentencias **READ** continúen admitiendo datos desde un valor concreto. Su formato general es:

RESTORE

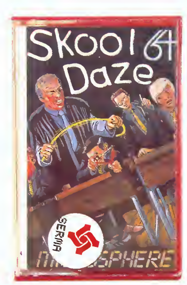
Un segundo método para forzar que las sentencias **DATA** sean tomadas nuevamente desde su principio es volver a ejecutar el programa con **RUN.** Esto se debe a que **RUN,** además de borrar el contenido de todas las variables, inicializa el puntero de lectura de las **DATA** al primer valor de la primera línea de ellas existente en el programa, para a continuación pasar a ejecutar el mismo a partir de la primera línea o de la especificada como argumento de la expresión.

Una de las aplicaciones de **RESTORE** puede ser la de facilitar una lectura repetitiva de determinado grupo de datos más de una vez dentro del programa. De esta forma, puede conseguirse la reconstrucción de los valores iniciales de las variables subindicadas, por ejemplo.



Una DATA permite suministrar como DATA una expresión que forzosamente deba ser evaluada, como por ejemplo una ecuación.

SKOOLDAZE



La lucha en los colegios entre profesores y alumnos ha estado siempre presente, la rebeldía del alumnado choca con la severidad de los maestros, y siempre se buscan profesores objeto de las burlas generales. Los exámenes son otro punto, puesto que están al día las «chuletas» y otros artilugios que el estudiante desarrolla con increíble habilidad, e incluso se llega al amaño en las correcciones y conocimiento de las preguntas anteriormente a la prueba.

En este caso concreto, tu eres Eric, terrible y travieso alumno de esos que no se están quietos ni un momento, y llevan a todos los profesores por él camino de la amargura. Algo malo has hecho, ya que hay un informe en la caja fuerte del colegio que debe llegar próximamente a la mesa del director, lo cual sería, sin duda, tu perdición. La única forma de impedirlo es abriendo la caja de seguridad y cambiando el informe. ¿Parece fácil?, pues no lo es, ya que posee una combinación de cuatro letras, de las cuales cada maestro conoce una, y la del director siempre va en primer lugar.

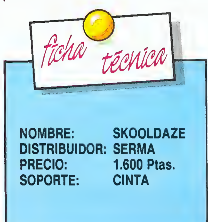
La única forma de lograr que cada uno de ellos te diga su combinación es golpeando las placas de acero que hay en las paredes de las clases, dado que reflejarán la luz hacia los ojos del maestro, desorientándole y logrando así que confiese la letra en cuestión. Por supuesto, unas placas serán fáciles de golpear, pero otras no, en estas deberás saltar, o incluso derribar a un compañero con tu tirachinas y subirte sobre él cuando caiga al suelo, pudiendo saltar más alto aún.

La mayor dificultad surgirá con el profesor de historia, que es el más anciano y su vista no anda ya muy bien, por lo cual es inmune a tus argucias. La única forma de conseguir su secreto será averiguar la fecha de su nacimiento, y escribirla en la pizarra cuando entre en clase; al verla, quedará hipnotizado y te dirá lo que quieres saber.

Si has conseguido la combinación entera, deberás buscar su orden correcto, pudiendo ya abrir la caja y cambiar el informe.

Ten cuidado, porque la escuela tiene sus reglas, tales como la prohibición de entrar en el despacho del director, o usar tirachinas, o correr por los pasillos; infringir alguna de estas reglas te costará el tener que escribir un gran número de líneas con la misma frase, si te descubren. Cuando tu lista de castigos sea muy grande te resultará casi imposible llevar a cabo tu misión, pues estarás más vigilado de lo normal.

Ya lo sabes Eric, eres el perfecto niño travieso y matón de la clase, ¡duro con ellos!



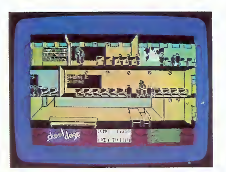
ficha técnica

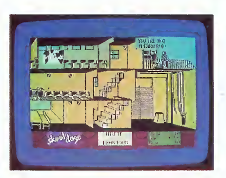
NOMBRE: SKOOLDAZE
DISTRIBUIDOR: SERMA
PRECIO: 1.600 Ptas.
SOPORTE: CINTA



veredicto final

GRAFICOS: ★★★
SONIDO: ★★★
INTERES: ★★★
DIFICULTAD: ★★★★
ORIGINALIDAD: ★★★★★



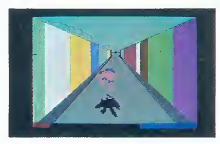


SUPERMAN

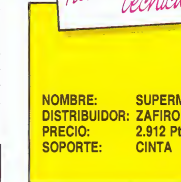


Llegó como todos sabemos, procedente de Krypton, planeta de diferentes condiciones al nuestro, lo cual le dió facultades sobrehumanas, como la vista infrarroja, capacidad de volar y su increíble fuerza y rapidez. ¿Cuántas veces vamos a repetir que no es un pájaro, ni un avión, sino que es... ¡TACHAAAANN!... ¡SUPERMAN!





Después de una vida placentera, salvando gatitos atrapados en árboles y niños que van a tocar los enchufes de la pared, Superman se va a tener que enfrentar con su más terrible enemigo, un loco científico maestro de la electrónica... Darkseid. Así, nuestro héroe deberá emprender de nuevo su encarnizada lucha contra el Mal, para preservar el Bien, la Verdad y la Justicia (¿dónde hemos oido eso antes?).



NOMBRE:	SUPERMAN
DISTRIBUIDOR:	ZAFIRO
PRECIO:	2.912 Ptas.
SOPORTE:	CINTA



GRAFICOS:	★★★★
SONIDO:	★★★
INTERES:	★★★★
DIFICULTAD:	★★★
ORIGINALIDAD:	★★★★★

La principal baza de Darkseid es su vista, que puede lanzar devastadores rayos Omega. Además, ha construido grandes guaridas subterráneas donde enclaustra a humanos con el fin de conocer sus identidades y acabar por encontrar a ciertas personas que poseen la fórmula Anti-vida en su mente; estas personas desconocen al mismo tiempo su posesión de la fórmula. Por ello, Superman debe salvar a los humanos de esos laberintos subterráneos e intentar debilitar a Darkseid mediante su visión calorífica.

Ambos lucharán en diferentes lugares de Metrópolis, dividida en seis sectores: tres bajo tierra y tres en la superficie. Incluso para pasar de una superficie a otra, Superman deberá superar las zonas de combate, en las cuales tendrá que impedir que Darkseid lance bombas sobre la ciudad, o volar a través de un largo pasadizo en el que Darkseid le atacará con rayos lasers, que podrían debilitar a nuestro héroe, hacer frente a la persecución del maligno, o incluso defenderse de cortinas de bolas de fuego en un estrecho espacio.

Durante los sucesivos encuentros, los oponentes se irán debilitando, debido a la lucha que mantienen, será muy importante recibir pocos impactos, puesto que la muerte significará la destrucción del mundo... y el fin del juego. En las calles de la ciudad, ambos contendientes se encontrarán con deflectores, que dirigirán la dirección de sus rayos dependiendo de la posición en que son colocados. Podremos elegir el papel de Darkseid (el malo), o de Superman (el bueno), para disputarnos el futuro de Metrópolis; todo depende de nuestras inclinaciones hacia uno u otro lado de la ley.

OUTLAWS

En el salvaje Oeste la vida transcurre entre dificultades, la ley todavía no está perfectamente estructurada y, por desgracia, los actos de pillaje, caciquismo y asesinatos son muy normales, mientras se espera que algún día llegue aquel que haga imperar la justicia.

Más allá de las llanuras de los indios y las grandes montañas del Este, un pequeño pueblecito creció gracias al árduo trabajo de sus gentes y conforme pasaron los años se fueron constituyendo en una colonia próspera y feliz, hasta el día en que llegaron los Proscritos, una de tantas bandas de pistoleros dedicadas al robo, saqueo y asesinato.

El grupo de facinerosos instaló su lugar de fechorías en la región y por ello, nadie podía estar seguro de vivir al día siguiente, viéndose obligados a pagar dinero para su protección, además de dar el fruto de su trabajo a los bandidos. En todas las familias faltaba algún miembro, «eliminado» por negarse a pagar o intentar resistirse a sus pretensiones.



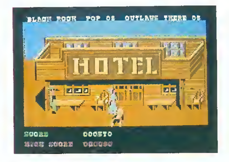


La gente vivió aterrorizada un largo tiempo, esperando que apareciese alguien lo suficientemente fuerte y valiente como para hacerles frente, y una noche oscura, como presagiando una época de tempestades, llegó el Jinete Solitario. Nadie le conocía ni le había visto nunca, pero en su cara se reflejaba la venganza y el odio contra aquellos que tenían bajo su yugo a la región.

De este modo, el Jinete Solitario se dedicó a perseguirlos sin tregua, a través de territorios inexplorados, de llanuras y desiertos, llegando a los terrenos de los indios y pequeñas ciudades de los alrededores, sin descanso.

Ahora eres tu el Jinete Solitario y deberás defenderte de los ataques de los asesinos; a lomos de tu caballo recorrerás praderas, desiertos, llanuras y poblados, intentando acabar con ellos; pero ten cuidado, puesto que ellos también poseen monturas y te atacarán a traición, en los lugares más insospechados, o simplemente escondidos tras las ventanas de una casa, contigo en el exterior, casi indefenso. Sólo te salvará de una muerte segura tu rapidez de disparo y reflejos.

Conforme vayas limpiando una región de bandidos irás pasando a las siguientes, en cada una la dificultad será mayor y el paisaje en el que te encontrarás, diferente. Se cauto y... veloz con el gatillo.







¡FUERA ERRORES!

C-Byte tiene el honor de presentar, en exclusiva nacional para los lectores de TU MICRO COMMODORE, el revolucionario sistema de introducción de programas FUERA ERRORES. Este nos permitirá introducir, sin temor alguno al esfuerzo inútil, cualquier listado por largo y complicado que parezca.

Para adoptar los listados publicados bajo este sistema, deberemos seguir las siguientes normas:

1) Es fundamental transcribir EXACTAMENTE el listado reproducido, incluyendo todos sus espacios, aunque se trate de separaciones entre número de instrucción y línea de instrucción.

2) Todas las líneas finalizarán con un número de tres dígitos, encerrado entre guiones, que NO deberá ser introducido, puesto que no forma parte del programa, sino que tiene la finalidad de hacer funcionar el sistema FUERA ERRORES, según veremos más adelante. Para evitar equivocaciones, dicha cifra entre guiones se sitúa en el margen derecho del final de la línea BASIC a la cual corresponde, a una distancia prudencial del mismo.

3) Para facilitar la introducción de símbolos difícilmente interpretables, se procede a la siguiente representación en los listados.

— Las letras aparecidas entre menor y mayor deberán ser introducidas con pulsación simultánea de la tecla COMMODORE y la letra representada. Ej.: <M> =COMMODORE M.

— Las letras aparecidas entre barras verticales deberán ser introducidas como pulsación simultánea de la tecla SHIFT y la letra representada. Ej.: ¦ K ¦ = SHIFT K.

— Entre corchetes simples se representarán los símbolos que se obtienen por pulsación directa de la tecla, aunque lógicamente, este caso sólo se dará para indicar las sucesiones de más de una letra. Así por ejemplo, la introducción de 5 asteriscos se representaría por [5*].

— Para la repetición de símbolos obtenidos mediante las teclas COMMODORE o SHIFT, se seguirá una combinación de las tres normas anteriormente citadas. Así por ejemplo, la introducción de 10 símbolos COMMODORE H, se representaría por [<10 H>].

— Para evitar confusiones, cuando se utilice el sistema de representación de sucesiones de carácter, y éste sea un espacio, se utilizará la abreviatura ESP. [15 ESP] = 15 espacios.

— Los carácteres de control, tales como desplazamientos del cursor, colores, estados de reversa y funciones, se simbolizarán por una abreviatura de tres letras (dos más un espacio en el caso de las funciones) encerrada entre llaves, tal como se señala en la tabla adjunta.

Para introducir cualquier listado por el sistema FUERA ERRORES, deberemos entrar previamente y ejecutar el listado BASIC que aparece en la página siguiente por lo cual es recomendable conservar una copia grabada del mismo, para sucesivas ocasiones. Una vez introducido este listado, ya sea por el teclado, o a través de cinta o disco, debemos ejecutarlo con **RUN.** Instantes más tarde aparecerá en la pantalla el mensaje **FUERA ERRORES!** y el cursor libre para la entrada de programas, con el tradicional **READY.** por encima, lo cual indicará la activación del sistema de depuración de errores.

En virtud al **NEW** que finaliza la línea 20 del programa ¡FUERA ERRORES!, este habrá desaparecido de la memoria, y seremos libres para introducir cualquiera de los programas listados en la sección **TECLA A TECLA** de cualquier número de nuestra revista, o incluso aquellos que apareciendo en otras secciones se acojan a este sistema. Así pues, si el programa ¡FUERA ERRORES! ha desaparecido de la memoria, ¿qué hemos conseguido ejecutándolo? Bien, la respuesta se llama informáticamente **INTERRUPCION:** se trata de una técnica de programación en código máquina que permite que el ordenador efectúe prácticamente dos trabajos a un tiempo, o más correctamente, que ejecute determinada tarea de forma automática, sin necesidad de que le prestemos una atención constante, de forma similar al proceso de respiración en un humano.

Efectivamente, aunque el soporte BASIC ha desaparecido de la memoria, antes de «marcharse» ha dejado funcionando en modo interrupción la pequeña rutina en código máquina que se hallaba en sus **DATAS.** Para comprobarlo pulsa **RETURN**; observarás algo muy extraño: tu ordenador no se comporta normalmente, no sólo desciende una línea el cursor, sino que además hace aparecer un número en la esquina superior izquierda de la pantalla. A continuación veremos como emplearlo.

Cada vez que pulsemos **RETURN**, aparecerá un número en la mencionada zona de la pantalla, y este corresponderá con la instrucción que hayamos introducido. Esto forma parte del sistema de FUERA ERRORES. Cuando introduzcamos cualquier línea de un listado de este tipo, deberemos fijarnos en el número que aparece al pulsar el **RETURN** de fin de línea; si este coincide con el que aprece en el listado al final de la línea, ésta habrá sido introducida correctamente, en caso contrario existe algún error de teclado que debemos modificar. Para modificar una instrucción errónea, no tenemos ni tan siquiera que volver a teclearla si no queremos, bastará sencillamente con modificar el carácter o caracteres erróneos como siempre hacemos, hasta que nos coincida el número de verificación que se presentará al pulsar **RETURN.**

Así pues, el sistema FUERA ERRORES se compone de dos partes: una codificación especial de los listados que facilitan su introducción, evitando los errores al confundir los caracteres gráficos, de control, etc... y un sistema de verificación de líneas que nos advierte en el preciso instante de introducir una de estas, que está mal tecleada.

Ahora bien, el empleo de estos dos sistemas no quiere decir que nos encontremos ante un BASIC diferente al de COMMODORE 64. Este no ha cambiado, hace exactamente las mismas cosas de siempre; simplemente hemos cambiado la forma de hacer los listados. En cuanto al misterioso número que aparece en la esquina de la pantalla no es más que



una simple suma de comprobación, lo que se conoce técnicamente como un CHECKSUM. La rutina en código máquina de interrupción suma los valores de los caracteres que entramos en cada línea que introducimos, aplicándoles un módulo 256, es decir, volviendo a cero cada vez que su suma parcial supera el 255, de este modo se crea un número entre 0 y 255 dependiente directamente de los caracteres introducidos. Así pues, es prácticamente imposible que una sucesión de errores den por casualidad ese número, mientras que siempre que la línea esté correctamente introducida obtendremos el mismo código que nosotros hallamos por igual sistema y añadimos al final de cada línea al realizar el listado del programa.

Por tanto, el misterioso número no es tampoco una modificación del BASIC de COMMODORE, sino simplemente un pequeño truco para la comprobación de que las líneas han sido bien introducidas. Propiamente no nos evita cometer errores de tecleado, sino que simplemente nos advierte inmediatamente en que línea los hemos introducido.

Para desactivar el sistema sólo deberemos pulsar RUN/STOP RESTORE, y si por cualquier motivo nos interesara reactivarlo, podríamos ejecutar SYS 822, siempre y cuando se encuentre el código máquina en la memoria, lógicamente.

¡ADVERTENCIA! Puesto que el código máquina se encuentra ubicado en el buffer del casete, es imprescindible desactivarlo (RUN/STOP RESTORE) antes de realizar cualquier operación con dicho periférico.

UTILIZACION DEL CASETE CON FUERA ERRORES

Si queremos introducir parte de un programa, para continuar posteriormente el trabajo emprendido, sin perder por supuesto la enorme ventaja del FUERA ERRORES deberemos proceder de la siguiente forma:

1. Desactivar el sistema FUERA ERRORES mediante la pulsación de RUN/STOP + RESTORE.
2. Trasladar el código máquina desde su ubicación en el buffer del casete hasta otro punto desocupado en la memoria RAM (por ejemplo 49152). Para ello sólo hemos de saber que ocupa 114 bytes desde 822 (inclusive). Una sencilla rutina que efectúe el trabajo de reubicación puede ser: FORI=0TO113:POKE49152+I,PEEK(822+I):NEXT.
3. Realizar la operación correspondiente con el casete; ya sea grabar una copia de seguridad de lo introducido en la memoria hasta el momento, ya sea cargar del casete una parte ya grabada del programa.
4. Restablecer FUERA ERRORES a su punto original. Siguiendo el caso de la línea de ejemplo anterior: FORI=0TO113:POKE822+I,PEEK(49152+I):NEXT.
5. Reactivar el sistema FUERA ERRORES mediante SYS 822 (esta vez no aparecerá mensaje alguno, aunque al pulsar **RETURN** comprobaremos que aparecen las cifras de control en la esquina superior izquierda). Lógicamente, los pasos 4 y 5 no son necesarios si después de una grabación se va a apagar el ordenador, y sólo se llevarán a cabo después de las cargas, o si tras la grabación de una copia de seguridad del programa, deseamos continuar introduciéndolo acto seguido.

TABLA DE INTERPRETACION DE CODIGOS DE CONTROL

ABR	SIFNIFICADO	OBTENCION	ABR	SIFNIFICADO	OBTENCION
HOM	HOME	CLR/HOME	F8	FUNCION 8	SHIFT F7
CLR	CLEAR + HOME	SHIFT CLR/HOME	BLK	BLACK (NEGRO)	CTRL 1
ABJ	CURSOR ABAJO	CRSR VERTICAL	WHT	WHITE (BLANCO)	CTRL 2
ARB	CURSOR ARRIBA	SHIFT CRSR VERTICAL	RED	RED (ROJO)	CTRL 3
DCH	CURSOR DERECHA	CRSR HORIZONTAL	CYN	CYAN (CIAN)	CTRL 4
IZQ	CURSOR IZQUIERDA	SHIFT CSRS HORIZONTAL	PUR	PURPPLE (PURPURA)	CTRL 5
RON	REVERSE ON	CTRL 9	GRN	GREEN (VERDE)	CTRL 6
ROF	REVERSE OFF	CTRL 0	BLU	BLUE (AZUL)	CTRL 7
F1	FUNCION 1	F1	YEL	YELLOW (AMARILLO)	CTRL 8
F2	FUNCION 2	SHIFT F1	NRJ	NARANJA	COMMODORE 1
F3	FUNCION 3	F3	MRN	MARRON	COMMODORE 2
F4	FUNCION 4	SHIFT F3	RCL	ROJO CLARO	COMMODORE 3
F5	FUNCION 5	F5	GR1	GRIS 1	COMMODORE 4
F6	FUNCION 6	SHIFT F5	GR2	GRIS 2	COMMODORE 5
F7	FUNCION 7	F7	VCL	VERDE CLARO	COMMODORE 6
			ACL	AZUL CLARO	COMMODORE 7
			GR3	GRIS 3	COMMODORE 8

```
10 FORI=822TO935:READA:C=C+A:POKEI,A:NEXT
20 IFC<>15254THENPRINTCHR$(147)"ATENCION!, HAY UN ERROR EN LOS DATOS":END
30 PRINTCHR$(147)TAB(213)"FUERA ERRORES!":SYS822:NEW
100 DATA 169,3,141,37,3,169,69,141,36,3,169,0,133,254,96,32,87,241,133,251
110 DATA 134,252,132,253,8,201,13,240,13,24,101,254,133,254,165,251,166,252
120 DATA 164,253,40,96,169,13,32,210,255,165,214,141,176,3,206,176,3,169,0
130 DATA 133,216,169,18,32,210,255,169,19,32,210,255,169,45,32,210,255,166
140 DATA 254,224,100,176,5,169,48,32,210,255,224,10,176,5,169,48,32,210,255
150 DATA 169,0,133,254,32,205,189,169,45,32,210,255,173,176,3,133,214,76,88,3
```

ST

¿Es udtes didlessico? ¿le fhalta koordinazion en los dedos? ¿es hincapaz de dhistigir dos numheros iquales? El programa STOP le da la oportunidad de demostrar lo contrario (¿o no?).



Desde Majadahonda, y la hache intercalada no va de broma —es que es el nombre de un pueblo de la provincia de Madrid— nos remite este programa Alvaro del Pozo Morato con destino al primer Concurso de Programación. Aunque no ha resultado afortunado con ningún premio, puesto que sus competidores en el certamen han aportado programas de mayor calidad, el suyo presenta ciertas características que le hacen digno de ser publicado, tal como se expresa en el punto 8 de las bases de nuestro concurso.

UN PROGRAMA ENORMEMENTE SENCILLO, MUY FACIL DE TECLEAR Y REALMENTE ENTRETENIDO DE JUGAR

El juego es muy simple. Al ejecutar el programa, aparecerán en el centro de la pantalla dos números en caracteres gigantes (para personas con miopía superior a las 9,5 dioptrías), separados mediante dos puntos (puntazos, más bien).

Estos números irán siendo alterados a velocidad considerable, y de una forma totalmente aleatoria, que como todos sabemos ya

OP

VERSION COMMODORE 64

```
1 REM ********** STOP ***********                       -249-
2 REM ********** POR   ***********                       -197-
3 REM ***** ALVARO DEL POZO****                          -019-
5 POKE 53281,0:POKE53280,0:PRINT"{WHT}{CLR}"
                                                         -109-
30 N$(1)=" |C||I| {ABJ}{4 IZQ}[2 ESP]|B| {ABJ}{4 IZQ}
[2 ESP]|B| {ABJ}{4 IZQ}[2 ESP]|B| {ABJ}{4 IZQ} |C| E
|C|{4 ARB}"                                              -217-
40 N$(2)="|U||2 C||I|{ABJ}{4 IZQ}[3 ESP]|B|{ABJ}{4 IZ
Q}|U||2 C||K|{ABJ}{4 IZQ}|B|[3 ESP]{ABJ}{4 IZQ}|J||2
C||K|{4 ARB}"                                            -251-
50 N$(3)="|U||2 C||I|{ABJ}{4 IZQ}[3 ESP]|B|{ABJ}{4 IZ
Q} |2 C| W {ABJ}{4 IZQ}[3 ESP]|B|{ABJ}{4 IZQ}|J||2 C|
|K|{4 ARB}"                                              -048-
60 N$(4)="|U|[3 ESP]{ABJ}{4 IZQ}|B|[2 ESP]|B|{ABJ}{4
IZQ} Z |2 C| W {ABJ}{4 IZQ}[3 ESP]|B|{ABJ}{4 IZQ}[3 E
SP]|E {4 ARB}"                                           -104-
70 N$(5)="|U||2 C||I|{ABJ}{4 IZQ}|B|[3 ESP]{ABJ}{4 IZ
Q}|J||2 C||I|{ABJ}{4 IZQ}[3 ESP]|B|{ABJ}{4 IZQ}|J||2
C||K|{4 ARB}"                                            -244-
80 N$(6)="|U||2 C||I|{ABJ}{4 IZQ}|B|[3 ESP]{ABJ}{4 IZ
Q} Q |2 C||I|{ABJ}{4 IZQ}|B|[2 ESP]|B|{ABJ}{4 IZQ}|J|
|2 C||K|{4 ARB}"                                         -121-
90 N$(7)="|U||2 C||I|{ABJ}{4 IZQ}[3 ESP]|B|{ABJ}{4 IZ
Q}[2 ESP]|C| W {ABJ}{4 IZQ}[3 ESP]|B|{ABJ}{4 IZQ}[3 E
SP]|E {4 ARB}"                                           -139-
100 N$(8)="|U||2 C||I|{ABJ}{4 IZQ}|B|[2 ESP]|B|{ABJ}{
4 IZQ} Q |2 C| W {ABJ}{4 IZQ}|B|[2 ESP]|B|{ABJ}{4 IZQ
}|J||2 C||K|{4 ARB}"                                     -048-
110 N$(9)="|U||2 C||I|{ABJ}{4 IZQ}|B|[2 ESP]|B|{ABJ}{
4 IZQ}|J||2 C| W {ABJ}{4 IZQ}[2 ESP]|B|{ABJ}{4 IZQ}|J
||2 C||K|{4 ARB}"                                        -175-
120 N$(0)="|U||2 C||I|{ABJ}{4 IZQ}|B|[2 ESP]|B|{ABJ}{
4 IZQ}|B|[2 ESP]|B|{ABJ}{4 IZQ}|B|[2 ESP]|B|{ABJ}{4 I
ZQ}|J||2 C||K|{4 ARB}"                                   -010-
130 P$="[3 ESP]{ABJ}{3 IZQ} |Q| {ABJ}{3 IZQ}[3 ESP]{A
BJ}{3 IZQ} |Q| {ABJ}{3 IZQ}[3 ESP]{4 ARB}"               -207-
140 FOR M=0 TO 9                                         -146-
150 LET B=INT(9*RND(1))                                  -063-
200 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
                                                         -176-
210 PRINTTAB(14);N$(M);P$;N$(B)                          -007-
212 FOR R= 1 TO 90                                       -232-
215 GET W$: IF W$="P"THEN 250:IF W$<>"P"THEN 280
                                                         -201-
216 NEXT                                                 -248-
218 IFM=B THEN 290                                       -032-
220 PRINT"{10 ARB}":NEXT                                 -168-
230 GOTO 140                                             -163-
250 IF M=B THEN 295                                      -065-
255 IF M<>B THEN 297                                     -133-
280 PRINT "TECLA EQUIVOCADA"                             -022-
281 GOTO 300                                             -167-
290 PRINT"NO PARO A TIEMPO"                              -202-
291 GOTO 300                                             -168-
295 PRINT "ACERTO"                                       -111-
296 GOTO 300                                             -173-
297 PRINT"NO ERAN IGUALES"                               -160-
300 END                                                  -138-
```

no tiene nada que ver con las alas, sino con el «tun-tun».

En el momento en que los dos números representados sean iguales (lo cual demostraría que sabemos cuando dos números son iguales), deberemos apresurarnos en pulsar la tecla **P** de australoPiteco (y también de Parar), con lo cual se detendrá el «baile» de números.

Si en el momento del arresto (detención) ambos números eran gemelos (iguales), habremos conseguido nuestro objetivo (ganar). Nihil obstat (No obstante), si no lo fueran (no números, sino iguales), habríamos perdido (pringado).

Así mismo, el programa, que es cantidad de astuto, también contempla la posibilidad de que nos equivoquemos al pulsar la tecla (sólo para disléxicos).

Como el juego así planteado sería excesivamente sencillo, se ha introducido una nueva dificultad, con la cual pondremos a prueba nuestros reflejos y coordinación dactilar: existe un brevísimo lapso de tiempo para distinguir la igualdad de los números y pulsar la tecla. Transcurrido dicho tiempo (o su lapso ese), el programa te declarará perdedor, con la emisión del mensaje NO PARO A TIEMPO (¡y es que el tal Tiempo venía embalado!).

Por si el juego no nos parece aún lo suficientemente complicado para nuestras cabecillas, se nos ocurren las siguientes sugerencias para dificultarlo paulatinamente (no de Pablo, sino de poco a poco).

1. Reducir la longitud del **FOR** de la línea 212.
2. Jugar a una distancia mínima de 10 metros del ordenador.
3. Pulsar la tecla mediante el lanzamiento de algún objeto arrojadizo: una botibota, un periférico, un hermano...
4. Jugar con las manos atadas a la espalda o bien con los ojos vendados o el televisor apagado.

Y en fin, ya está bien de dar ideas, que luego se acostumbran y...

UNA ULTIMA HABILIDAD DEL PROGRAMA: SU COMPATIBILIDAD CON C-16, C-64, Y C-128

Para hacer extensivo este test sobre dislexia, etc... a todos los lectores de nuestra revista, el único, el inigualable, el fabuloso, el fantástico JE-FE DE RE-DAC-CI-ON de nuestra revista (si después de esto no nos sube el sueldo...) ha tenido a bien dignarse a compatibilizar el programa (aplausos), para el Commodore 128. He aquí el sistema.

Utilícese el aparato en modo C-64.

Posteriormente ha ordenado a alguno de sus esclavos que realice la compatibilización correspondiente para C-16 (el se encontraba cansado debido al esfuerzo intelectual efectuado con anterioridad). Como resultado de esta última acción de mando, publicamos el listado correspondiente.

I lla nada mas ha teklear.

VERSION COMMODORE 16

```
1 REM *** STOP VERSION C-16 ***
2 REM *** F. LOPEZ MARTINEZ ***
3 REM *** IDEA ORIGINAL DE: ***
4 REM **** ALVARO DEL POZO ****
5 COLOR4,1,0:COLOR0,1,0:PRINT"{WHT}{CLR}"
30 N$(1)=" !C!!I! {ABJ}{4 IZQ}[2 ESP]!B! {ABJ}{4 IZQ}
[2 ESP]!B! {ABJ}{4 IZQ}[2 ESP]!B! {ABJ}{4 IZQ} !C!!E>
!C!{4 ARB}"
40 N$(2)="!U!!2 C!!I!{ABJ}{4 IZQ}[3 ESP]!B!{ABJ}{4 IZ
Q}!U!!2 C!!K!{ABJ}{4 IZQ}!B![3 ESP]{ABJ}{4 IZQ}!J!!2
C!!K!{4 ARB}"
50 N$(3)="!U!!2 C!!I!{ABJ}{4 IZQ}[3 ESP]!B!{ABJ}{4 IZ
Q} !2 C! W {ABJ}{4 IZQ}[3 ESP]!B!{ABJ}{4 IZQ}!J!!2 C!
!K!{4 ARB}"
60 N$(4)="!U![3 ESP]{ABJ}{4 IZQ}!B![2 ESP]!B!{ABJ}{4
IZQ} Z !2 C! W {ABJ}{4 IZQ}[3 ESP]!B!{ABJ}{4 IZQ}[3 E
SP] E {4 ARB}"
70 N$(5)="!U !2 C!!I!{ABJ}{4 IZQ}!B![3 ESP]{ABJ}{4 IZ
Q}!J!!2 C!!I!{ABJ}{4 IZQ}[3 ESP]!B!{ABJ}{4 IZQ}!J!!2
C!!K!{4 ARB}"
80 N$(6)="!U!!2 C!!I!{ABJ}{4 IZQ}!B![3 ESP]{ABJ}{4 IZ
Q} Q !2 C!!I!{ABJ}{4 IZQ}!B![2 ESP]!B!{ABJ}{4 IZQ}!J!
!2 C!!K!{4 ARB}"
90 N$(7)="!U!!2 C!!I!{ABJ}{4 IZQ}[3 ESP]!B!{ABJ}{4 IZ
Q}[2 ESP]!C! W {ABJ}{4 IZQ}[3 ESP]!B!{ABJ}{4 IZQ}[3 E
SP] E {4 ARB}"
100 N$(8)="!U!!2 C!!I!{ABJ}{4 IZQ}!B![2 ESP]!B!{ABJ}{
4 IZQ} W !2 C! W {ABJ}{4 IZQ}!B![2 ESP]!B!{ABJ}{4 IZQ
}!J!!2 C!!K!{4 ARB}"
110 N$(9)="!U!!2 C!!I!{ABJ}{4 IZQ}!B![2 ESP]!B!{ABJ}{
4 IZQ}!J!!2 C! W {ABJ}{4 IZQ}[3 ESP]!B!{ABJ}{4 IZQ}!J
!!2 C!!K!{4 ARB}"
120 N$(0)="!U!!2 C!!I!{ABJ}{4 IZQ}!B![2 ESP]!B!{ABJ}{
4 IZQ}!B![2 ESP]!B!{ABJ}{4 IZQ}!B![2 ESP]!B!{ABJ}{4 I
ZQ}!J!!2 C!!K!{4 ARB}"
130 P$="[3 ESP]{ABJ}{3 IZQ} !Q! {ABJ}{3 IZQ}[3 ESP]{A
BJ}{3 IZQ} !Q! {ABJ}{3 IZQ}[3 ESP]{4 ARB}"
140 FORM=0TO9
150 B=INT(9*RND(1))
200 FORI=0TO7:PRINT:NEXT
210 PRINTTAB(14)N$(M)P$N$(B)
212 FORR=1TO90
215 GETW$:IFW$="P"THEN250
216 IFW$<>"P"ANDW$<>""THEN280
217 NEXT
218 IFM=BTHEN290
220 PRINT"{10 ARB}":NEXT
230 GOTO140
250 IFM=BTHEN295
255 IFM<>BTHEN297
280 PRINT"{HOM}TECLA EQUIVOCADA"
281 GOTO300
290 PRINT"{HOM}NO PARO A TIEMPO"
291 GOTO300
295 PRINT"{HOM}ACERTO"
296 GOTO300
297 PRINT"{HOM}NO ERAN IGUALES"
300 END
```

GRAFICOS DE BARRAS VERTICALES

Aparte de las características más relevantes de nuestros ordenadores Commodore, a saber, sonido, sprites, alta resolución, etc.; hay una que quizá no se utilice con demasiada frecuencia: los caracteres o símbolos preprogramados.

Todos hemos visto esos curiosos caracteres presentes en el teclado, y que se acceden por medio de SHIFT o COMMODORE en modo mayúsculas, y al final descubrimos que sólo sirven para trabajar en modo baja resolución. Pues bien, la rutina que presentamos hoy, tiene por objetivo generar gráficos de barras utilizando la facilidad de los caracteres preprogramados.

En modo normal (baja resolución), nuestro Commodore nos ofrece una resolución de 25 líneas de 40 columnas cada una, así pues, si quisiéramos representar algo gráficamente en este modo, y eligiésemos la representación en vertical, nos veríamos obligados a obtener una resolución máxima de 25 puntos. Por otra parte, podemos realizar el trabajo en alta resolución, con lo que obtendríamos una resolución 8 veces superior, es decir, de 200 puntos o pixels (25×8), pero ya sabemos por propia experiencia lo difícil que es trabajar en este modo; así pues, ¿qué podemos hacer? En el párrafo siguiente encontraremos la solución.

Aquí es donde vienen en nuestra ayuda los gráficos definidos. Afortunadamente, tomando el caracter en sentido vertical (8 pixels al igual que en horizontal), disponemos de los suficientes caracteres para representar cualquier altura (de 0 a 8) que deseemos, si bien no todos ellos están disponibles en modo normal, sino que algunos se deben utilizar en modo reverso.

El programa que presentamos a continuación, es una demostración del uso de esta subrutina; para ello habrá que tener en cuenta lo siguiente:

La altura de la barra será entre 0 y 160, y se entregará en la variable Y.

En la variable X (1–35) ubicaremos la posición horizontal de comienzo de la barra.

El color de la barra lo determina la variable CO.

El programa de demostración consiste en las líneas 1–9 (construcción de la pantalla) y línea 100 (simulación de un caso real). En las líneas 200-260 se encuentra la rutina propiamente dicha, que podremos utilizar con nuestros propios programas. La línea 240 determina el ancho de la barra (0: un caracter, 1: dos caracteres, etc.). Hemos de tener cuidado para no salirnos de la pantalla al hacer uso de barras demasiado anchas, lo mejor es utilizarlas de un caracter o dos.

Los usuarios del C16 deberán modificar las siguientes líneas del listado:

```
2FORI=0TO7:READC(I):
NEXT:COLOR1,6:COLOR0,1
100 Y=40:FORX=1TO35STEP3:
CO=CO+1:Y=Y+9:GOSUB200:
NEXT110 GOTO 110
230 E=INT (Y/8):CR=Y-E*8
250  P=3915+H:Q=-1024:FORI=1
TOE:POKEP+X,160:
POKEP+X+Q,CO:P=P-40:NEXT
```

```
1 DATA 32,100,111,121,98,248,247,227                        -035-
2 FORI=0TO7:READC(I):NEXT:POKE53281,0:POKE646,5
                                                            -167-
3 PS$="ESTADISTICA DE INGRESOS":PRINT"{CLR}"TAB((40-L
EN(PS$))/2);PS$                                             -004-
4 PRINT"{2 ESP}{37 @}";                                     -116-
5 FORI=9TO0STEP-1:I$=MID$(STR$(I*10),2):IFI=0THENI$="
00"                                                         -055-
6 PRINT"{3 ESP}{M}{35 ESP}{G}";                             -110-
7 PRINT I$;"{M}{35 @}{G}";:NEXT                             -124-
8 PRINT"{3 ESP}{37 T}";                                     -093-
9 PRINT"{4 ESP}E{2 ESP}F{2 ESP}M{2 ESP}A{2 ESP}M{2 ES
P}J{2 ESP}J{2 ESP}A{2 ESP}S{2 ESP}O{2 ESP}N{2 ESP}D";
                                                            -020-
100 Y=40:FORX=1TO35STEP3:CO=CO+1:Y=Y+9:GOSUB200:NEXT:
SYS42115:REM END                                            -160-
200 IFY>160THENY=160                                        -203-
210 IFX<1 OR X>35 THENPRINT"X FUERA DE RANGO":STOP
                                                            -177-
220 IF(COAND15)=0THENCO=CO+1                                -184-
230 ER=INT(Y/8):CR=Y-ER*8                                   -016-
240 FORH=0TO1                                               -038-
250 P=1867+H:Q=54272:FORI=1TOER:POKEP+X,160:POKEP+X+Q
,CO:P=P-40:NEXT                                             -001-
260 POKEP+X,C(CR):POKEP+X+Q,CO:NEXT H:RETURN
                                                            -129-
```

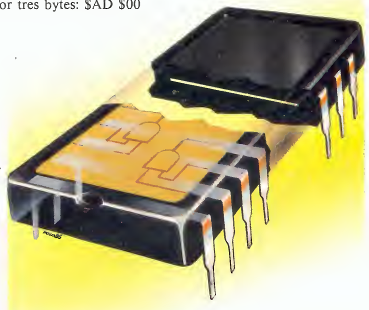
ENSAMBLADORES Y CODIGO MAQUINA: TAL PARA CUAL

Muchas veces, al hablar de programación en código máquina realmente se está hablando de programación en ensamblador. El ensamblador es lo que nos permite programar el C/M con más facilidad: estudiémoslo.

El código máquina puro, tal como lo entiende el microprocesador, no es más que una sucesión de números. Para cualquiera de nosotros, preparar un programa en C/M sería muy pesado, y en cuanto se alargase un poco, imposible.

Un ejemplo sencillo nos convencerá definitivamente: llevar al acumulador el contenido de la posición de memoria $1000 se representa por tres bytes: $AD $00 $10; si a esto le añadimos que la CPU maneja números binarios, el C/M «puro» será: 10101101 00000000 00010000, ante lo cual sobra cualquier comentario...

Para ayudarnos en la programación en código máquina, se han desarrollado los programas ensambladores. Un esamblador es, por decirlo de alguna manera, un compilador que convierte un código alfanumérico más comprensible en lenguaje máquina puro.



Una instrucción de código máquina consta de un byte para la instrucción propiamente dicha, la cual indica la operación que deseamos realizar (leer, escribir, sumar...) y el modo de direccionamiento que utiliza, y de un OPERANDO, que es el dato sobre el cual actuamos (bien un dato o una posición de memoria). Este puede constar de uno, dos o incluso ningún byte, dependiendo del modo de direccionamiento de la instrucción.

En ensamblador, cada operación viene representada por un código mnemónico, una abreviatura de tres letras que nos recuerda la utilidad de la instrucción. Así, por ejemplo, la instrucción «cargar en el acumulador» se representa por LDA, abreviatura de LoaD Accumulator (cargar acumulador; en inglés, ya se sabe...).

LOS OPERANDOS TAMBIEN SE SIMPLIFICAN SI UTILIZAMOS UN ENSAMBLADOR

Para representar los operandos existen diversas formas, según el tipo de que se trate. Veremos en

profundidad la representación de cada uno al hablar de los modos de direccionamiento, pero de momento repasemos algunas ideas básicas.

Para referirnos a direcciones de memoria, simplemente debemos escribirlas tras el código de la operación, en hexadecimal generalmente (aunque algunos ensambladores permiten el decimal).

Un detalle que nos permite programar con mas facilidad es que, en vez de referirnos a direcciones, podemos asignar dichas direcciones a un nombre simbólico, y utilizar este en el programa. Así, si nuestro programa guarda en la posición de memoria $1000 los puntos de un juego, podemos escribir LDA PUNTOS en lugar de LDA $1000, siempre y cuando hayamos definido previamente el valor de la VARIABLE PUNTOS, de la forma que estudiaremos más adelante.

El nombre de «variable» puede inducir a error. No es lo que en BASIC entendemos, es decir, un número sobre el cual operamos, sino más bien una constante que empleamos para referirnos cómodamente a cierta posición de memoria durante el uso del ensamblador.

Cuando queremos ejecutar, durante el programa, una bifurcación a una cierta instrucción, podemos operar de forma similar al caso anteror: referirnos a la instrucción con un nombre simbólico, que se denomina comúnmente ETIQUETA. Debemos situar antes de la instrucción destino el nombre de la etiqueta. De esta forma, el ensamblador, busca por el programa hasta encontrarla, y calcula el salto que debe dar la CPU, ahorrándonos a nosotros este ingrato trabajo.

Además del propio programa, el ensamblador necesita de datos adicionales, como la dirección de memoria en la cual debe comenzar el programa, o la asignación de los nombres simbólicos.

Para esto se utilizan las pseudoinstrucciones, que no son en realidad parte del programa, sino órdenes directas al propio ensamblador. Estas pseudoinstrucciones encabezan por lo general el programa.

Existe una gran variedad de ellas, propias de cada ensamblador, debido a lo cual lo mejor es consutar el manual del nuestro. Una de las pseudoinstrucciones más importantes es el asterisco, que representa la dirección donde deseamos que comience el programa. Así: * = $1000 indica que nuestro programa comienza en la posición $1000. Es bastante frecuente que se emplee el mnemónico ORG en lugar del asterisco, pero su función es la misma.

UNA VEZ COMPLETADO EL PROGRAMA, EL ENSAMBLADOR LO CONVIERTE EN CODIGO MAQUINA

Cuando hemos terminado de escribir el programa, podemos darle al ensamblador la orden de ensamblarlo (traducirlo a C/M). Existen ensambladores que traducen directamente las instrucciones al introducirlas, pero son las menos.

Si al realizar el ensamblado se detecta algún tipo de error, el ensamblador lo comunicará, generando un mensaje apropiado. Si el ensamblado se realiza correctamente, podremos volcar en un periférico (generalmente en disco) el contenido del área de memoria escrita.

En el listado adjunto podemos ver un ejemplo típico de un listado producido por un ensamblador. Distinguiremos en él cuatro zonas, de izquierda a derecha, después de la dirección de la instrucción: La ZONA DE ETIQUETAS, donde se sitúan éstas; la ZONA DE OPERACIÓN, en la cual se ubica el nmemónico de la instrucción; la ZONA DE OPERANDOS, en la que se encuentran estos; y finalmente la ZONA DE COMENTARIOS, en la cual podemos escribir, tras un punto y coma, frases o comentarios que nos ayuden más tarde a comprender el funcionamiento de programa, de modo similar a los REM del BASIC.

Con estas indicaciones, podemos comprender el funcionamiento de un ensamblador e inciarnos en el camino hacia el cerebro del micro.

```
DIR$            CODE  OPER.     COMMENTS

0000            * =   $1000     ; origen en $1000
1000            LDA     $20     ; carga espacio en acumulador
1002            LDX     $C9     ; carga X para 5 líneas
1004  BUCLE     STA   $03FF,X   ; guarda en pantalla
1007            DEX             ; decrementa X
1008            BNE   BUCLE     ; si no es cero vuelve al bucle
100A            RTS             ; retorno
ERRORS = 00000
END OF ASSEMBLY
```

CONCURSO

El travieso C-Byte tiene el honor de invitaros a la participación en nuestro primer concurso de programación. Los requisitos necesarios son bien pocos:

- Saber programar un ordenador COMMODORE.
- Ser español o extranjero y
- Tener una edad comprendida entre 5 y 105 años.

Fácil, ¿verdad?
En cuanto a los premios, la mar de atractivos:

- **1.er premio.–60.000 pesetas en material informático a escoger por el premiado, más un póster de C-Byte.**
- **2.° premio.–30.000 pesetas en material informático a escoger por el premiado, más un póster de C-Byte.**
- **3.er premio.–15.000 pesetas en material informático a escoger por el premiado, más un poster de C-Byte.**

Y en fin, si alguno de los programas destaca por su originalidad, estética o comicidad, no sería de extrañar que le cayera alguna cosilla más...

Las bases del concurso son bien sencillas:

1. Los programas remitidos al concurso deberán ser creación original del autor o autores, y completamente inéditos, pudiendo remitir tantos programas como se desee.

2. Los programas deberán ser enviados en casette o disco flexible a TU MICRO COMMODORE (Concurso de programación). Apartado de correos 61.294. 28080 MADRID.

3. Los programas podrán ser de cualquier tipo (juegos, utilidades, gestión, educativos) y habrán de estar escritos en lenguaje BASIC o código máquina.

4. Los programas deberán ser remitidos desprovistos de cualquier tipo de protección, que impida o dificulte el análisis del mismo, así como reproducción en estas páginas y su introducción como listado siguiendo el sistema FUERA ERRORES.

5. Cuando la ejecución del programa precise de la concurrencia de determinado periférico o aditamento (joysticks, tabletas gráficas, programas comerciales de ayuda), se valorará la indicación de las modificaciones pertinentes, para que el programa pueda ser disfrutado por cualquier usuario en la configuración básica.

6. Todo programa presentado al concurso deberá acompañarse de los siguientes datos:
   - Datos personales del concursante
   - Nombre del programa.
   - Modelo para el que está destinado.
   - Breve descripción del programa detallando las indicaciones necesarias para su ejecución.

7. Los programas premiados pasarán a ser propiedad de la revista TU MICRO COMMODORE, pudiendo hacer ésta libre uso de ellos, y renunciando sus autores a cualquier otra compensación distinta al premio.

8. Los programas no premiados que por su calidad se hagan merecedores de su publicación, serán adquiridos por la editorial, aplicando la tarifa vigente.

9. Los programas recibidos con posterioridad a la fecha tope de la presente edición, serán automáticamente incluidos en los destinados a la siguiente.

10. El jurado decidirá sobre todos los aspectos no contemplados en estas bases y su decisión será inapelable.

Y ahora a darse mucha prisa, el plazo para la recepción de programas termina el próximo día 15 de marzo.

¡¡SUERTE!!







La revista semanal donde el usuario de COMMODORE podrá encontrar gran variedad de programas y noticias, sus periféricos y equipos.

SUSCRIPCION POR TELEFONO

Todos los días, excepto sábados y festivos, de **9 a 15,00 y de 17 a 19,00 horas** atenderemos sus consultas en el ☎ 250 58 20/250 55 79.
¡Suscríbase por teléfono! y en pocos días tendrá en su domicilio TU MICRO COMMODORE.

SUSCRIPCION POR CORREO

Deseo suscribirme a la revista TU MICRO COMMODORE al precio de 7.400 ptas. **(incluido IVA),** por el período de un año a partir del N.º....... inclusive.

El importe **—que abonaré en su totalidad con el primer envío—** lo haré efectivo de la siguiente forma:

☐ Talón bancario adjunto a nombre de INGELEK, S. A. ☐ Giro Postal n.º ....
☐ Tarjeta VISA n.º ____ ____ ____ ____ FIRMA
☐ Tarjeta MASTER CARD n.º ____ ____ ____ ____
Fecha de caducidad de la tarjeta .... Nombre titular ....

APELLIDOS ______ NOMBRE ______
DOMICILIO ______ EDAD ______
CIUDAD ______ PROVINCIA ______
C. POSTAL ______ TELEFONO ______ PROFESION ______

Recorte o copie esta tarjeta y envíela dentro del sobre a: Ediciones INGELEK
Apartado de Correos 61.294. 28080 MADRID





Su Commodore 64 tiene mucho que decirle.

Monitores.

El Commodore 64 es el resultado de la experiencia internacional de Commodore como líder indiscutible en el mercado de los microordenadores.

El Commodore 64 es el ordenador más completo y potente de su categoría,... pero todavía tiene mucho que decirle.

Por ejemplo, sus Monitores.

Monitores funcionales de diseño con pantalla de alta resolución.



Pensados y creados para trabajar con ellos sin notar molestias ni cansancio en la vista.

Monitores capaces de reproducir con nitidez la completa gama de colores del C-64.

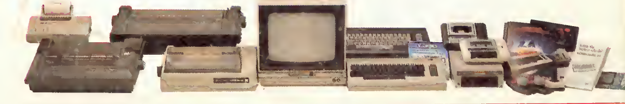
Monitores con un cuidado sistema de sonido. Con conexiones conmutables de Vídeo Compuesto y Crominancia-Luminancia.

Monitores que complementan las prestaciones de su ordenador, ganando en imagen.

Amplíe las posibilidades de su C-64, descubriendo su extensa gama de periféricos.

Ahora que ya sabe que su Commodore 64 tiene todavía mucho que decirle, prepárese a conocerle mejor.

commodore



Microelectrónica y Control, S.A. *c/ Valencia, 49-53 08015 Barcelona - c/ Ardemans, 8 28028 Madrid.*
Unico representante de Commodore en España.